UNITED STATES PATENT APPLICATION FOR:

Title: Multiprocessor-Scalable Streaming Data Server Arrangement

Inventors:

Deep K. BUCH
Zhenjun HU
Neil SCHAPER
David ZHAO
Vladimir PENTKOVSKI

Prepared by:

Antonelli, Terry, Stout & Kraus, LLP
1300 North Seventeenth Street, Suite 1800
Arlington, Virginia 22209
Tel: 703/312-6600
Fax: 703/312-6666
E-mail: hzykorie@antonelli.com

Title: Multiprocessor-Scalable Streaming Data Server Arrangement

FIELD

[0001] The present invention is directed to streaming data server arrangements. More particularly, the present invention is directed to a multiprocessor-scalable streaming data server arrangement.

BACKGROUND

[0002] One application area for servers is to provide streaming data over the Internet or over a corporate intranet. The streaming data may consist of multimedia data (e.g.-audio/video), character streams (e.g.-stock quotes), etc. On-demand content refers to the streaming of specific client-requested data files whereas broadcast content refers to content delivered from a source, such as an encoder system, onto an incoming network link, and streamed out to thousands of clients over a set of outgoing network links.

[0003] It is been found through measurements that currently used multiprocessor servers have poor SMP (Symmetric Multi-Processing) scalability. In most cases, the processor is the performance limiter. In such processor-bound cases using processors having 2 MB L2 caches, the following was noted:

[0004] A very high CPI (Clocks per Instruction retired) was noted with measured CPI ranges of between 4.0 and 6.0, which is considerably above the average for most server applications.

[0005] A very high L2 MPI (Level 2 Cache Misses per Instruction retired) was noted with measured L2 MPI's in the range of 2% to 4%. This indicates that on the average, 3 out of every 100 instructions results in an L2 miss.

[0006] A saturated front-side bus was noted. That is, performance counters show that the data bus is actively transferring data 40% of the time. When accounting for the read/write transaction mix, bus efficiency, and MP arbitration, this indicates that the front-side bus is close to being saturated.

[0007] The raw data bandwidth requirements for streaming are typically much lower than the capabilities of the system and the I/O buses. Thus, the observed saturation of the bus clearly indicates that there is a large overhead of unnecessary data transfers.

[0008] It has been found that poor scalability of such systems are due to the following factors:

[0009] Interrupt/DPC (Deferred Procedure Call) Migration: Hardware interrupts from the NIC (Network Interface Cards) are routed to any available processor by the OS (Operating System). The DPC handler, which is set up by the OS, is executed in turn by some other processor.

[0010] Loosely-coupled Connection Processing: Client connections are processed by different processors during connection lifetimes. The same processors process both input and output streams.

[0011] Thread and Buffer Migration: The threads of the server process are not bound to any specific processor. Thus, during the course of transferring data between input and output buffers, the server thread runs on different processors at different times. This migration of threads leads to the pollution of the processor caches, since the same buffer ping-pongs around between processors.

[0012] Inefficient L2 Cache Utilization: The large L2 processor caches are not properly utilized. The nature of streaming data is non-temporal, that is, the data is used only once and never used again, thus, the caching of this data serves no useful purpose

and yet the data is loaded into the L2 caches. Since the threads write to the data buffers in order to extract/append network protocol headers, this leads to dirty write-backs when the next incoming buffer is accessed by a processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The foregoing and a better understanding of the present invention will become apparent from the following detailed description of example embodiments and the claims when read in connection with the accompanying drawings, all forming a part of the disclosure of this invention. While the foregoing and following written and illustrated disclosure focuses on disclosing example embodiments of the invention, it should be clearly understood that the same is by way of illustration and example only and that the invention is not limited thereto. The spirit and scope of the present invention are limited only by the terms of the appended claims.

[0014] The following represents brief descriptions of the drawings, wherein:

[0015] Figure 1 illustrates an example arrangement of a multi-processor scalable streaming data server.

[0016] Figure 2 illustrates tightly-coupled connection processing, interrupt/DPC binding, and thread/buffer affinity.

[0017] Figure 3 illustrates improved L2 cache utilization using SSE (Streaming SIMD Extensions).

[0018] Figure 4 illustrates an example arrangement of a multi-scalable streaming data server in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0019] Before beginning a detailed description of the subject invention, mention of the following is in order. When appropriate, like reference numerals and characters may be used to designate identical, corresponding, or similar components in differing drawing figures. Furthermore, in the detailed description to follow, example sizes/models/values/ranges may be given, although the present invention is not limited thereto. Arrangements may be shown in block diagram form in order to avoid obscuring the invention and also in view of the fact that specifics with respect to implementation of such block diagram arrangements are highly dependent upon the platform within which the present invention is to be implemented, that is, such specifics should be well within the purview of one skilled in the art. Where specific details are set forth in order to describe example embodiments of the invention, it should be apparent to one skilled in the art that the invention can be practiced without, or with variations of, the specific details. Still furthermore, well-known elements have been omitted from the drawing figures so as not to obscure the invention. Lastly, it should be apparent that differing combinations of hard-wired circuitry and software instructions can be used to implement embodiments of the present invention, that is, the present invention is not limited to any specific combination of hardware and software.

[0020] Figure 1 illustrates an example arrangement of a multi-processor scalable streaming data server. As shown in Figure 1, a chipset 120 and memory 130, having both in input buffer IN_BUF and an output buffer OUT_BUF, are connected via an I/O bus to a plurality of network interface cards NIC0-NIC3 and are also connected via a front-side bus to a plurality of processors CPU0-CPU3. The plurality of processors each have respective L2 caches 110-113.

[0021] As shown in Figure 1, in step 1, incoming streaming data is transferred to the input buffer IN_BUF by the network interface card NIC0, the incoming streaming data, for example, consisting of an incoming media stream from a media encoder system 100. In step 2, a server thread A on processor CPU2 reads data out of the in buffer IN_BUF. In step 3, the server thread A is next scheduled to run on processor CPU1, resulting in an example of thread/buffer migration. In step 4, the server thread A copies data to output buffer OUT_BUF, resulting in an example of inefficient L2 cache utilization. In step 5, network output card NIC3 generates a hardware interrupt which is serviced by an ISR (Interrupt Service Routine) running on the processor CPU3 and a DPC callback is registered. In step 6, the DPC handler is executed on processor CPU2, resulting in interrupt/DPC migration. In step 7, the DPC handler invokes server thread B, which runs on processor CPU0. In step 8, the server thread B completes packet assembly and marks output buffer OUT_BUF ready for network input card NIC3 to read. Lastly, in step 9, the network card NIC3 reads data out of the output buffer OUT_BUF and streams the data to the clients.

[0022] Thus, the disadvantageous example arrangement of Figure 1 suffers from poor scalability due to the four factors noted above in the Background. To improve scalability, the present invention includes one or more of the following features:

[0023] Interrupt/DPC Binding: In the present invention, the number of network interface cards is equal to the number of processors which is equal to N. Interrupts from the network interface card that handles the incoming data stream are bound to a single processor CPU0. Interrupts for network interface card NICn are bound to respective processor CPUn where 0 < n < = N. The DPC for network interface card NICn is also

bound to processor CPUn where 0 < = n < = N. This obviates the interrupt/DPC migration problem.

[0024] Tightly-coupled Connection Processing: In the present invention, given a set of client connections C0-CM, the client connections are tightly coupled to specific processors except for processor CPU0 which is reserved for input stream processing. For example, connections C0 to CM/(N -1) are assigned to processor CPU1, etc. This obviates the loosely-coupled connection processing problem.

[0025] Thread and Buffer Affinity: In the present invention, given a set of server threads T0 to Tp which access and process streaming data buffers, these server threads are bound to specific processors except for processor CPU0 (which is reserved for input stream processing). This is effected using thread affinity API calls. For example, in Windows, the SetThreadAffinityMask() call may be used. For example, threads T0 to Tp/(N -1) are bound to processor CPU1, etc. This also ensures that buffers corresponding to a thread are processed only by the processor to which the thread is bound. This obviates the thread and buffer migration problem.

[0026] Efficient L2 Cache Utilization with SSE Non-temporal Prefetch and Streaming Store Instructions: In the present invention, with regard to instructions, since the code for the data server threads does not change, the instructions can be allowed to be cached in the L2 caches of the processors. With regard to temporal data, section data associated with client connections is temporal and can be allowed to be cached in the L2 caches of the processors. With regard to non-temporal data, streaming data is non-temporal and is therefore not cached in the L2 caches of the processors. Rather, the data is inputted to the lowest level cache, that is, the L1 cache, using, for example, the prefetchnta instruction to the SSE. Packet assembly is effected in the L1 cache and

the finished packet is inputted to the memory buffer using, for example, the movntq streaming store instruction outputted by the MMx registers. This obviates the inefficient L2 cache utilization problem.

[0027] Efficient L1 Cache Utilization: In the present invention, the L1 cache efficiency may be improved by increasing the amount of time allotted to the server threads which process streaming buffers.

[0028] Figure 2 illustrates the tightly-coupled connection processing, interrupt/DPC binding, and thread/buffer affinity features of the present invention. As illustrated Figure 2, network interface card NIC0, which is bound to processor CPU0, along with L2 cache 110, are reserved for input stream processing of data from the media encoder system 100. The processor CPU0 has its server thread, DPC, and ISR bound thereto to ensure thread and buffer affinity. Similarly, processors CPU1-CPU3 are bound to respective network interface cards NIC1-NIC3 to ensure interrupt/DPC binding. The processors CPU1-CPU3 also have their respective server threads, DPC's, and ISR's bound thereto. Furthermore, the network interface cards NIC1-NIC3 are respectively dedicated to specific client connections to ensure tightly-coupled connection processing.

[0029] Figure 3 illustrates improved L2 cache utilization using SSE. As illustrated in Figure 3, a processor 300 includes a level 1 (L1) cache and an MMx register as well as a level 2 (L2) cache. Data from the input buffer IN_BUF is inputted to the level 1 cache and data is output from the MMx register to the output buffer OUT_BUF. Data is transferred between the output buffer OUT_BUF and a network interface card 350 via a Bus-master DMA. As shown in Figure 3, the non-temporal streaming data completely bypasses the L2 cache. The prefetchnta instruction is used to input the data directly

from the input buffer IN_BUF into the L1 cache. This is followed by a regular movq instruction to load the data into an MMx register. Then, the movntq instruction is used to output data from the MMx register into the output buffer OUT_BUF. This sequence of steps may be disposed within a software loop to copy the streaming data directly from the input buffer IN_BUF to the output buffer OUT_BUF. The number of words moved per loop iteration can be optimized for a particular processor microarchitecture.

[0030] The L2 cache is used to store instructions and temporal data, such as session state and connection management information. The UDP and IP headers for outgoing packets are prefixed to the data in the output buffer OUT_BUF in the usual way.

[0031] Figure for illustrates an example arrangement of a multi-scalable streaming data server in accordance with the embodiment of the present invention. As illustrated in Figure 4, the various elements can be easily seen as corresponding to their respective elements in Figure 1. In step 1 of Figure 4, incoming streaming data is transferred to the input buffer IN_BUF of the memory 430 by the network interface card NIC0. In step 2, the server thread A of the processor CPU0 reads data out of the input buffer IN_BUF using a non-temporal prefetch (bypassing the L2 cache 410). In step 3, the server thread A copy's data to the output buffer OUT_BUF, (a streaming store which bypasses the L2 cache 410 and therefore results in efficient L2 cache utilization). In step 4, the network interface card NIC3 generates a hardware interrupt which is serviced by the processor CPU3 and the DPC callback is registered. In step 5, the DPC handler actually executes on the processor CPU3 (interrupt/DPC binding). In step 6, the DPC handler invokes server thread B which runs on processor CPU3 (server thread/buffer affinity). In step 7, the server thread B completes packet assembly and locks the output buffer OUT_BUF ready for the network interface card NIC3 to read. Lastly, in

step 8, the network interface card NIC3 reads data out of the output buffer OUT_BUF and streams it to the clients connected thereto.

[0032] The impact of the above-noted features of the present invention is that the bottlenecks that limit scalability are effectively removed. Front-side bus utilization is greatly reduced by limiting the overheads of buffer migration. Cache efficiency also improves due to greatly reduced L2 MPI and the processing efficiency, as measured by CPI, also improves. Since each processing unit (consisting of a processor and its respective network interface card) is almost independent, the multiprocessor scalability also improves significantly.

[0033] This concludes the description of the example embodiment. Although the present invention has been described with reference to an illustrative embodiment thereof, it should be understood that numerous other modifications and embodiments can be devised by those skilled in the art that will fall within the spirit and scope of the principles of this invention. More particularly, reasonable variations and modifications are possible in the component parts and/or arrangements of the subject combination arrangement within the scope of the foregoing disclosure, the drawings, and the appended claims without departing from the spirit of the invention. In addition to variations and modifications in the component parts and/or arrangements, alternative uses will be apparent to those skilled in the art.

What is claimed is: